Picture languages and locality

Simo Tukiainen

Department of Mathematics and Statistics
Faculty of Science
University of Helsinki

June 2020

HELSINGIN YLIOPISTO — HELSINGFORS UNIVERSITET — UNIVERSITY OF HELSINKI

Tiedekunta/Osasto — Fakultet/Sektion — Faculty	Laitos — Institution — Department
Faculty of Science	Department of Mathematics and Statistics

Tekijä — Författare — Author
Simo Tukiainen

Työn nimi — Arbetets titel — Title
Picture languages and locality

Oppiaine — Läroämne — Subject
Mathematics

Työn laji — Arbetets art — Level	Aika — Datum — Month and year	Sivumäärä — Sidoantal — Number of pages
Master's thesis / pro gradu	June 2020	46 pages

Tiivistelmä — Referat — Abstract

In this master's thesis we study the generalization of word languages into multi-dimensional arrays of letters i.e picture languages. Our main interest is the class of recognizable picture languages which has many properties in common with the robust class of regular word languages. After surveying the basic properties of picture languages, we present a logical characterization of recognizable picture languages—a generalization of Büchi's theorem of word languages into pictures, namely that the class of recognizable picture languages is the one recognized by existential monadic second-order logic. The proof presented is a recent one that makes the relation between tilings and logic clear in the proof. By way of the proof we also study the locality of the model theory of picture structures through logical locality obtained by normalization of EMSO on those structures. A continuing theme in the work is also to compare automata and recognizability between word and picture languages. In the fourth section we briefly look at topics related to computativity and computational complexity of recognizable picture languages.

Avainsanat — Nyckelord — Keywords
Picture languages, logic, finite model theory, recognizable picture languages, EMSO, automata

Säilytyspaikka — Förvaringsställe — Where deposited
Kumpula Campus Library

Muita tietoja — Övriga uppgifter — Additional information

Contents

Acknowledgements

I am grateful to Professor Juha Kontinen for suggesting this interesting topic and supervising the work with much patience. I am also indebted to Dr Antti Kuusisto for reviewing the thesis on such a short notice.

I also thank my parents and Tomoko for always supporting me.

Simo Tukiainen
Helsinki, June 2020

Introduction

Regular languages are a very robust class of word languages, as testified by their multiple conceptually different yet equivalent characterizations and their powerful closure properties. In the present work, we look at a generalization of word languages to multi-dimensional arrays that are commonly called pictures and especially the class of recognizable picture languages, which in many ways generalizes the class of regular languages of words. We also present a generalization of Büchi's theorem to picture languages and show that the class of recognizable picture languages defined by tilings coincides with the class of picture languages definable in existential monadic second-order logic.

Recall that the class of regular word languages can be equivalently characterized as recognized

- by deterministic finite automata
- by non-deterministic finite automata
- by existential monadic second-order logic (Büchi's theorem)
- by regular expressions (Kleene's theorem)
- algebraically using equivalence relations (Myhill–Nerode's theorem).

However, while pictures as multi-dimensional arrays are a rather obvious generalization of word languages, it is not at all obvious how the class of regular word languages or word automata should be generalized to pictures. It turns out that straightforward attempts to generalize regular expressions or finite automata result in a class of picture languages that is unsatisfactory in many ways e.g one that is either too restricted or has poor closure properties. However, in [GR92] was introduced the class of recognizable word languages (REC), which has since received much interest and is generally thought of as the best generalization of regular word languages into pictures.

One of the attractions of recognizable picture languages is—like with regular word languages—its many different characterizations; REC can be characterized by tiling systems, online tessellation automata and by logic, to mention only few. In [GRST96] was introduced for picture languages the equivalent of Büchi's theorem of word languages—the result that the class of recognizable picture languages is the same as the class of picture languages definable in existential monadic second-order logic (EMSO). We dedicate the third section to the study of this characterization by presenting a more recent proof from [GO16], which—in addition to generalizing the proof to any finite dimension instead of two—offers an alternative point of view by proving

the theorem through a syntactical normalization of EMSO on the class of structures for pictures, which in turn gives the reader much information about the logic of pictures and eventually makes the relation to characterization by tilings obvious. In the fourth section we informally discuss some additional properties of recognizable picture languages related to computativity and computational complexity.

Throughout this paper, first-order logic is used with its usual semantics. When talking about logic, concepts *formula*, *sentence*, *atomic sentence* and *term* have their usual meanings. A *literal* is either an atomic sentence or its negation—the former being called *positive literal* and the latter *negative literal*. A *clause* is a disjunction of literals. By EMSO we mean the existential monadic second-order logic with sentences of the form

$$\exists U_1 \ldots \exists U_n \, \psi$$

where $U_1, \ldots U_n$ are unary predicate symbols and ψ is a first-order sentence. By EMSO($\forall^1$) we mean the fragment of EMSO where each sentence is of the form

$$\exists U_1 \ldots U_n \forall x \, \psi(x)$$

with $\psi(x)$ a quantifier-free first-order formula with x as its only free variable. Whenever we write a formula in form $\psi(x_1, \ldots, x_n)$, the assumption is that all its free variables are among $x_1, \ldots, x_n$.

We tend to begin our enumerations from 1, so given a positive integer n we use $[n]$ to denote the set $\{1, 2, \ldots, n\}$.

1 Word languages

We will first review some basics of word languages. Most theorems in this section will be stated without proofs—a comprehensive treatment of descriptive complexity of word languages that covers every proof omitted in this section can be found in [Ebb95].

1.1 Basic definitions

Definition 1.1. An *alphabet* is a non-empty, finite set of symbols that we often call *letters*. Given an alphabet Σ, we write $\Sigma^\star$ for the set of finite sequences of its elements and call them *words*. We write λ for the empty sequence and call it the *empty word*. We denote the length of word w as $\mathrm{l}(w)$. A subset of $\Sigma^\star$ is called a Σ-*language* or a *language over* Σ.

The basic boolean operations of *union*, *intersection*, *complement* on languages are defined as the corresponding set operations, the complement being understood relative to $\Sigma^\star$. We also define some operations more particular to words:

Definition 1.2. Given words w_1 and w_2, $w_1 w_2$ is called the *concatenation* of the words and is defined as the joined sequence. The *concatenation of languages* L_1 and L_2 is written $L_1 L_2$ and defined as

$$L_1 L_2 = \{w_1 w_2 \colon w_1 \in L_1, w_2 \in L_2\}.$$

Definition 1.3. Given language L the *Kleene's star* operation is written $L^\star$ and is defined as

$$L^\star = \{w_1 \dots w_n \colon n \in \mathbb{N}, w_1, \dots, w_n \in L\}$$

i.e as the concatenation of finitely many elements of L. Note that this always includes the empty word.

1.2 Finite automata

Definition 1.4. Given an alphabet Σ, a *non-deterministic finite automaton (NFA)* is defined by a tuple (Q, q_0, δ, F), where

- Q is a finite set of states
- q_0 is the initial state
- $\delta \subseteq Q \times \Sigma \times Q$ is the transition relation

- $F \subseteq Q$ is the set of accepting states.

The automaton is *deterministic* (DFA) if δ is a partial function $Q \times \Sigma \to Q$ i.e the relation associates at most one state with each pair of $Q \times \Sigma$.

Definition 1.5. Given an automaton $M = (Q, q_0, \delta, F)$, we define the evaluation function

$$\tilde{\delta} \colon Q \times \Sigma^\star \to \mathrm{Pow}(Q)$$

inductively as

$$\begin{aligned} \tilde{\delta}(q, \lambda) &= \{q\} \\ \tilde{\delta}(q, w\alpha) &= \{p | (r, \alpha, p) \in \delta \text{ for some } r \in \tilde{\delta}(q, w)\} \end{aligned}$$

The automaton *accepts* word $w \in \Sigma^\star$ if $\tilde{\delta}(q_0, w) \cap F \neq \emptyset$. A Σ-language L is recognized by the automaton if $L = \{w \in \Sigma^\star \mid M \text{ accepts } w\}$.

Note that if the automaton is deterministic, $\tilde{\delta}(q, w)$ is always either the empty set or a singleton for any $(q, w) \in Q \times \Sigma^\star$, and $\tilde{\delta}$ can thus be considered a partial function $Q \times \Sigma^\star \to Q$; hence the determinacy—there is only one way the automaton can be evaluated for a given word. Note that whenever we wish to consider our transition function or evaluation function of a deterministic automaton a non-partial function, it is always possible to add one dummy non-accepting state to cover all the missing transitions. Thus we may assume them to be full functions for deterministic automata when it is convenient to do so.

The idea of finite automata is simpler than how its formal definition may seem at first: an automaton starts in the initial state and for each character of the input word a new state is chosen among the possible states given by the transition relation for the combination of the old state and the letter now being considered. Once the whole input word has been thus processed, the run is considered an accepting run if the last state reached—the state where the automaton stopped—is of one the accepting states. For a non-deterministic automaton, the word is accepted, if there exists an accepting run using any of the possible state transitions; non-accepting runs will often also exist in this case, but finding one accepting run is enough.

A valid run of the automaton implies a sequence of state transitions as can be seen form the inductive definition of the evaluation function. For accepting runs, the last state of the sequence is one of the accepting states. This intuitive idea is easier to use in many proofs than the definition of the evaluation function, so we formalize it in the following proposion and its corollary.

Proposition 1.1. *Given a non-deterministic automaton $M = (Q, q_0, \delta, F)$, states $q, r \in Q$ and a word $w = \alpha_1 \dots \alpha_n \in \Sigma^\star$, the following are equivalent*

- $r \in \tilde{\delta}(q, w)$
- *either w is the empty word and $q = r$ or otherwise there exists a sequence of transitions*

$$((q, \alpha_1, r_1), (r_1, \alpha_2, r_2), \dots, (r_{n-1}, \alpha_n, r_n)) \in \delta^n$$

such that $r_n = r$.

Proof. We will prove the claim by induction on the definition of the evaluation function $\tilde{\delta}$. If $w = \lambda$, the only claim that $q = r$ follows directly from the fact that we have $\tilde{\delta}(q, \lambda) = \{r\}$ as per definition of $\tilde{\delta}$.

Assume then that the claim holds for w and let $\alpha \in \Sigma$ be given. Assume first that $r \in \tilde{\delta}(q, w\alpha)$. If w is the empty word, $((q, \alpha, r))$ is our sequence. Otherwise there exists $r' \in$ such that $(r', \alpha, r) \in \delta$ and $r \in \tilde{\delta}(r', w)$. Hence by induction assumption there exists a sequence

$$((q, \alpha_1, r_1), \dots, (r_{n-1}, \alpha_n, r_n))$$

with $r' = r_n$. Now the sequence

$$((q, \alpha_1, r_1,), \dots, (r_{n-1}, \alpha_n, r_n), (r_n, \alpha, r))$$

is the sequence we wanted. The other direction is essentially the same. □

Corollary 1.1. *Given a non-deterministic automaton $M = (Q, q_0, \delta, F)$, a word $w = \alpha_1 \dots \alpha_n \in \Sigma^\star$ is accepted by the automaton iff either w is the empty word and $q_0 \in F$ or otherwise there exists a sequence of transitions*

$$((q_0, \alpha_1, r_1), (r_1, \alpha_2, r_2), \dots, (r_{n-1}, \alpha_n, r_n)) \in \delta^n$$

such that $r_n \in F$.

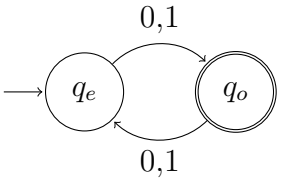
Proof. Follows directly from the definition of the automata accepting a word and the previous proposition. □

Example 1.1. The automata in figure 1a accepts the $\{0, 1\}$-language of words with odd length. It's formal definition would be (Q, q_0, δ, F) where $Q = \{q_e, q_o\}$, $q_0 = q_e$, $F = \{q_o\}$ and

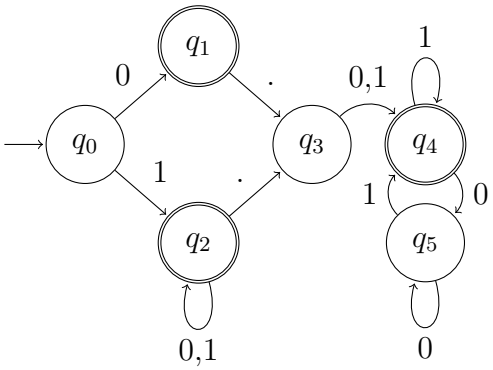
$$\delta(q_e, 0) = \delta(q_e, 1) = q_o$$
$$\delta(q_o, 0) = \delta(q_o, 1) = q_e.$$

The automaton starts in the "even" state, switches state between "even" and "odd" for each input character and accepts the word if it ends up in the "odd" state.

Figure 1: Examples of deterministic finite automata expressed as transition diagrams



(a) Words of odd length



(b) Numbers in binary decimal notation

Example 1.2. The automaton in figure 1b accepts the $\{0, 1, .\}$-language of binary decimal numbers without unnecessary leading or trailing zeroes. The initial state is q_0 and accepting states are q_1, q_2 and q_4. Note that while it accepts 0, 0.0, 1.0 and 1.1, it rejects e.g 01, 1.10 and .0 as well as non-zero integers with leading zeroes and any trailing zeroes. The intuitive meanings of for example non-accepting states q_3 and q_5 would be "decimal separator read, expecting to read one or zero" and "one or more trailing zeroes detected, waiting to read one", respectively.

The next theorem was proven in [RS59], where non-deterministic automata were also first introduced.

Theorem 1.1. *Given a language L over alphabet Σ, the following are equivalent:*

- *L is recognized by a deterministic finite automaton*
- *L is recognized by a non-deterministic finite automaton.*

As the deterministic and non-deterministic finite automata are equivalent for word languages, we mostly refer to them simply as *finite automata* from now on.

Reviewing further basic properties of finite automata, we have the following propositions.

Proposition 1.2. *The class defined by finite automata is closed under union, intersection and complement.*

Proposition 1.3. *The class defined by finite automata is closed under concatenation and Kleene's star.*

The propositions essentially show that the class of languages definable by finite automata is equivalent to the class regular languages, which is defined by regular expressions.

Next we define the model theoretic structures for words of a given alphabet. We will make no direct use of them in the present paper, but the definition will be useful for comparing it with the corresponding structure for pictures i.e pixel structures introduced in the third section.

Definition 1.6. Given an alphabet Σ and a word $w = \alpha_1 \dots \alpha_n \in \Sigma^\star$ with n a positive integer, the *word model* of w—denoted by $\mathcal{M}(w)$—is defined as the model $([n], <, (Q_\alpha)_{\alpha\in\Sigma})$, where $<$ is the natural ordering of $[n]$ and for $\alpha \in \Sigma$ and $i \in n$ we have $i \in Q_\alpha$ iff $\alpha_i = \alpha$.

Given a Σ-language L such that $\lambda \notin L$, we define word models of L as

$$\mathcal{M}(L) = \{\mathcal{M}(w) | w \in L\}.$$

A sentence ψ of a given logic is said to *define* language L if the models of the sentence are $\mathcal{M}(L)$, up to isomorphism.

We have excluded the world model for the empty word due to problems that arise in logic if empty models are allowed. As the recognizability of the empty word is a trivial aspect of the theory of automata, we can simply leave out languages that include the empty word. Alternatively we could had defined the definability of a language modulo the inclusion of the empty word.

Büchi's theorem, one the basic results of the theory of automata and important for introducing logical and model theoretic perspective into study of word languages was introduced in [Bü60].

Theorem 1.2. *Given a language L over alphabet Σ, the following are equivalent:*

- *L is recognized by a finite automaton*
- *L is defined by a sentence of* EMSO*.*

1.3 Two-way finite automata

Two-way deterministic (non-deterministic) finite automaton, also called 2DFA (2NFA), is a finite automaton with a head that can move in both directions i.e it need not perform only one scan of the word and decide the final state.

For the reader familiar with Turing machines, it will be good to know that it can equivalently be defined as a Turing machine that operates on a single read-only input tape i.e the machine cannot modify the contents of the tape but only change its own internal state.

On word languages, these characterizations collapse to simply that of ordinary finite automata as shown by the following theorem, which was proved in [RS59].

Theorem 1.3. *Given language L, the following are equivalent:*

- *L is recognized by a DFA*
- *L is recognized by a 2DFA*
- *L is recognized by a 2NFA.*

Two-way finite automata is interesting to us as its natural generalization to pictures is the four-way finite automata, that can make moves in four directions. Four-way automata will be introduced in section 2.4.

2 Picture languages

Many generalizations of ordinary word languages have been studied, for example ω-languages, where words have infinite length, or tree languages, where words have the structure of a tree. These constructs share many important aspects of the theory of word languages such as the equivalence between finite automata, non-deterministic finite automata and EMSO. In this section we will introduce the theory of picture languages as a generalization of word languages into two-dimensional pictures. Part of the theory is presented without proofs and many aspects are considered only informally. An overview of automata, regular expressions and logical characterizations of two-dimensional picture languages can be found in [GR97].

Two-dimensional languages provide yet another natural generalization of word languages and they are usually called pictures as a rectangular grid of coloured pixels is the natural representation of a picture e.g in computer science. It has, however, proven difficult to generalize regular expressions and automata to pictures so that they would retain some of their robust properties.

In the following, we assume Σ to be a finite alphabet.

Definition 2.1. A *picture* over Σ is a non-empty rectangular array of symbols of Σ, formally any function $p\colon [m] \times [n] \to \Sigma$ for some positive integers m, n. Given a picture $p\colon [m] \times [n] \to \Sigma$ we call the pair (m, n) its *size* and define $l_1(p) = m$ and $l_2(p) = n$ which we also call its *height* and *width* respectively.

A *subpicture* is a continuous rectangular block within a picture i.e a picture p is a subpicture of a picture q if for some positive integers i_1, j_1 with $l_1(p) + i_1 \leq l_1(q)$ and $l_2(p) + j_1 \leq l_2(q)$ it holds for all i, j with $i_1 \leq i \leq l_1(p)$ and $j_1 \leq j \leq l_2(p)$ that

$$p(i, j) = q(i_1 + i, j_1 + j)$$

Given positive integers h, k and a picture p with $l_1(p) \geq h$ and $l_2(p) \geq k$, we define the set of *tiles* having size (h, k), written as $T_{h,k}(p)$, as the set of all subpictures of p with size (h, k).

A set of pictures over Σ is called a Σ-*language* and the language of all pictures over Σ is denoted by $\Sigma^{\star\star}$.

For our tilings and automata we also need to be able to describe behaviour at the edges of a picture, so we define a bordering for our pictures.

Definition 2.2. We write $\Sigma^\sharp$ for alphabet $\Sigma \cup \{\sharp\}$ where $\sharp$ is a new special symbol called the *border symbol* that is not in Σ. Given a picture p with

Figure 2: Coordinate system for a picture p of size $(m, n) = (l_1(p), l_2(p))$

$p(1,1)$	$\cdots$	$p(1, l_2(p))$
$\vdots$	$\ddots$	$\vdots$
$p(l_1(p), 1))$	$\cdots$	$p(l_1(p), l_2(p))$

Figure 3: A picture p over alphabet $\{0, 1\}$ and its bordered version.

0	1	0
1	0	1
0	1	0

(a) picture p

♯	♯	♯	♯	♯
♯	0	1	0	♯
♯	1	0	1	♯
♯	0	1	0	♯
♯	♯	♯	♯	♯

(b) Bordered picture $p^\sharp$

size (m, n), we write $p^\sharp$ for its *bordered picture* of size $(m + 2, n + 2)$ over $\Sigma^\sharp$ defined as:

$$p^\sharp(i, j) = \sharp, \text{ when of } i = 1,\ i = m + 2,\ j = 1 \text{ or } j = n + 2$$
$$p^\sharp(i, j) = p(i - 1, j - 1), \text{ otherwise.}$$

Example 2.1. In figure 3 we have a picture p over a two-letter alphabet and its bordered version $p^\sharp$.

2.1 Regular expressions

We can define basic operations on pictures analogous to regular expressions of word languages. We notice that generalization of string concatenation is not trivial, because there are now at least two obvious ways to concatenate pictures i.e horizontally and vertically, given that they have the same height and width respectively. We begin by defining these basic concatenations.

Definition 2.3. Let p and q be pictures over Σ.
If the pictures have the same width i.e $l_2(p) = l_2(q)$, we define their *row concatenation* $p \ominus q$ as the picture of size $(l_1(p) + l_1(q), l_2(p))$ such that

$$(p \ominus q)(i, j) = p(i, j), \text{ when } 1 \leq j \leq l_1(p)$$
$$(p \ominus q)(i, j) = q(i - l_1(p), j), \text{ otherwise.}$$

Figure 4: Row and column concatenation operations on pictures p and q.

$p(1,1)$	$\cdots$	$p(1,l_2(p))$
$\vdots$	$\ddots$	$\vdots$
$p(l_1(p),1))$	$\cdots$	$p(l_1(p),l_2(p))$
$q(1,1)$	$\cdots$	$q(1,l_2(q))$
$\vdots$	$\ddots$	$\vdots$
$q(l_1(q),1))$	$\cdots$	$q(l_1(q),l_2(q))$

(a) Row concatenation $p \ominus q$

$p(1,1)$	$\cdots$	$p(1,l_2(p))$	$q(1,1)$	$\cdots$	$q(1,l_2(q))$
$\vdots$	$\ddots$	$\vdots$	$\vdots$	$\ddots$	$\vdots$
$p(l_1(p),1))$	$\cdots$	$p(l_1(p),l_2(p))$	$q(l_1(q),1))$	$\cdots$	$q(l_1(q),l_2(q))$

(b) Column concatenation $p \oplus q$

Similarly, if the pictures have the same height i.e $l_1(p) = l_1(q)$, we define their *column concatenation* $p \oplus q$ as the picture of size $(l_1(p), l_2(p) + l_2(q))$ such that

$$(p \oplus q)(i,j) = p(i,j), \text{ when } 1 \leq j \leq l_2(p)$$
$$(p \oplus q)(i,j) = q(i, j - l_2(p)), \text{ otherwise.}$$

Visualizations of row and column concatenations are shown in figures 4a and 4b, respectively.

Next we extend the above concatenation operations to whole languages, as well as define the other regular expression operations.

Definition 2.4. Given Σ-languages L_1 and L_2, we define

- their *union* and *intersection* as the corresponding set operations
- their *row concatenation*

$$L_1 \ominus L_2 = \{p_1 \ominus p_2 \colon p_1 \in L_1, p_2 \in L_2, l_2(p) = l_2(q)\}$$

- their *column concatenation*

$$L_1 \oslash L_2 = \{p_1 \oslash p_2 \colon p_1 \in L_1, p_2 \in L_2, l_1(p) = l_1(q)\}.$$

For a Σ language L, we define its *complement* L^c as $\Sigma^{\star\star} \setminus L$

Finally we define operations corresponding to Kleene star operator for pictures.

Definition 2.5. Given Σ-language, we iteratively define its

- *column closure* $L^{\star\oslash} = \bigcup_{i\in\mathbb{N}} L_i$ where $L_0 = L$ and $L_{i+1} = L \oslash L_i$
- *row closure* $L^{\star\ominus} = \bigcup_{i\in\mathbb{N}} L_i$ where $L_0 = L$ and $L_{i+1} = L \ominus L_i$.

One can now choose a set of above operations and study the class of languages generated from 1×1 pictures using the selected operations. However, none of the classes so formed seem to have as robust properties as the class of regular word languages—either their lack expressive power in not recognizing some simple pictures or they have poor closure properties. Furthermore, one indeed needs to include many of the operations as basic ones unlike for word languages; for example, the class of languages generated by unions and closure operators only—as are regular word languages—is not closed under intersection. For more detailed comparison between the classes of regular picture languages and recognizable picture languages—that are introduced in the next section—see [GR97] and [Mat07].

2.2 Recognizable languages

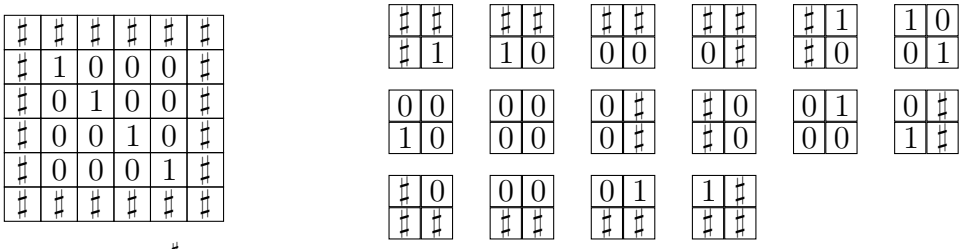
Recognizable two-dimensional languages were first defined in [GR92]. We begin by defining local languages using a finite set of allowed tiles that are allowed to occur in a picture. Computationally a local language can be described as a memoryless walk over a given picture where each tile given by a pixel's neighbourhood is inspected to see whether it is allowed or not.

Definition 2.6. A picture language L over Σ is *local* if there exists a set Δ of pictures over Σ with size $(2, 2)$ such that

$$L = \{p \in \Sigma^{\star\star} \colon T_{2,2}(p^\sharp) \subseteq \Delta\}$$

i.e L is exactly the set of pictures that include only tiles allowed by Δ.

Figure 5: One picture p with ones in diagonal and its tiles



(a) picture $p^{\sharp}_{\text{diag}}$

(b) The tile set $T_{2,2}(p^{\sharp}_{\text{diag}})$

Example 2.2. Figure 5 shows $p^{\sharp}_{\text{diag}}$ and its set of 2×2 tiles $T_{2,2}(p^{\sharp}_{\text{diag}})$. The picture p_{diag} is special in the way that the tile set of its bordered version can be used to define the language L_{diag} of square pictures over alphabet $\{0, 1\}$ with ones on diagonal i.e

$$L_{\text{diag}} = \{p \in \{0,1\}^{\star\star} \colon T_{2,2}(p) \subseteq T_{2,2}(p^{\sharp}_{\text{diag}}).\}$$

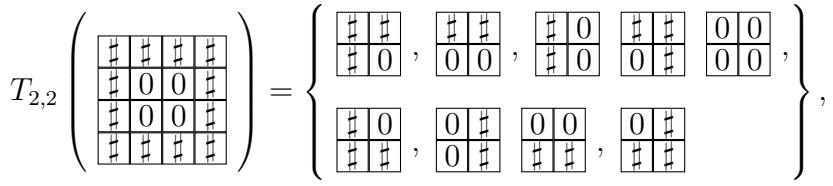
Note that p_{diag} is the smallest picture with this property. This shows that the language L_{diag} is local with $T_{2,2}(p^{\sharp}_{\text{diag}})$ as its set of allowed tiles.

Next we introduce projections that allow the use of extra symbols during recognition. This seemingly simple addition to local languages makes it very powerful—using the picture-walking analogy from above, it both allows for bounded memory per pixel and an ability to guess, which introduces non-determinacy.

Definition 2.7. A picture language L on Σ is *recognizable* if it is the projection of a local language over some alphabet Σ'. In other words, there exists a local language L' over Σ' and a function $\pi \colon \Sigma' \to \Sigma$ such that

$$L = \pi(L') = \{\pi \circ p \colon p \in L'\}.$$

Example 2.3. Let L_{square} be the language of squares over the one letter alphabet $\{0\}$ i.e the language of pictures p with $l_1(p) = l_2(p)$. It is easy to see that L_{square} cannot be local: the tiling would have to allow at least tiles



but this tiling already allows for any picture of $\{0\}^{\star\star}$. On the other hand, if we define a projection $\pi\colon \{0,1\} \to \{0\}$ as

$$\pi(0) = \pi(1) = 0,$$

we see that $\pi(L_{\text{diag}}) = L_{\text{square}}$, where L_{diag} is the language from example 2.2. Now, L_{diag} was shown previously to be local, so L_{square} is now found to be recognizable as a projection of a local language.

This result makes sense: we need extra markings to track certain features and drawing a line along the diagonal from corner to corner is an intuitively appealing way to verify that a rectangular array-like shape is square.

In [GRST96] it was shown that recognizable languages are exactly those definable in existential monadic second-order logic. In the third section we present a proof generalized to any finite dimension of pictures, which also proves this original theorem, which we present here without defining model theoretic structure for pictures yet.

Theorem 2.1. *Given a language L the following are equivalent:*

- *L is recognizable*
- *L is definable in* EMSO.

It was shown in [GR96] that REC is closed under regular expression operations save complementation.

Theorem 2.2. REC *is closed under:*

- *projection*
- *row and column concatenation*
- *row and column closure*
- *union and intersection*
- *rotation.*

Proof. See [GR97]. □

However, unlike in word languages, we do not have closure under complementation, as was first shown in [IT92].

Theorem 2.3. *The class of recognizable languages is* not *closed under complementation.*

Proof. See [GR97] for a simpler proof than [IT92]. The combinatorial argument is much like commonly used counting arguments for regular word languages: we introduce symmetry the keeping track of which would require unbounded counting abilities and then show that any tiling system and a projection that would recognize the language in question would also be forced to accept something not belonging to it (cf. pumping lemmas for word automata). □

So the class of recognizable picture languages has some attracting properties, but how to justify that it is a generalization of regular word languages in two dimensions when any of the commonly used definitions such as regular expressions or automata are so different? One more test to perform is to encode one-dimensional case in pictures somehow and show that the definitions yield the same result. To this end, we will first show that regular word languages are recognizable when encoded as pictures—which is not very remarkable—and afterwards show that word-like recognizable picture languages are regular as word languages, which is more interesting.

We use the following convention of encoding words as pictures.

Definition 2.8. Given a word w, we define its corresponding picture p_w of size $(1, \mathrm{l}(w))$ such that $p_w(1, i) = w_i$ for $1 \leq i \leq \mathrm{l}(w)$. We call a picture *word-like* when it has size $(1, n)$ for some positive integer n. A picture language is word-like when it only contains word-like pictures.

In the next two proofs, we will make use of the following fact.

Remark 2.1. For word-like recognizable languages, we may without any loss of generality assume that a 4×4 tile set T over Σ' has the following property: for any $a, b \in \Sigma'^{\sharp}$ it holds that $\begin{array}{|c|c|}\hline \sharp & \sharp \\ \hline a & b \\ \hline \end{array} \in T$ iff $\begin{array}{|c|c|}\hline a & b \\ \hline \sharp & \sharp \\ \hline \end{array} \in T$. We can make this assumption because any actual word-like picture will always include neither or both, so if a tiling included only one of them, removing it would still result in the same local language.

Due to this assumption, we can consider any given tiling for a word-like language to consist of tiles simply of form $\begin{array}{|c|c|}\hline a & b \\ \hline \end{array}$ or pairs (a, b) for $a, b \in \Sigma'^{\sharp}$. We also use this as a shorthand when defining a tiling for a word-like language and understand the necessary squares to be defined.

The next two propositions will establish a one-to-one correspondence between the class of word languages accepted by a non-deterministic finite automata and the class of recognizable picture languages. As we have not defined empty pictures, we simply ignore any automata that accept the empty

word; recall that an automaton accepts the empty word iff the initial state is included among the accepting states—this means that the question whether a given automaton accepts the empty word is not very interesting.

Proposition 2.1. *Given a regular word-language L_w over Σ such that $\lambda \notin L_w$, the language L_p defined as*

$$L_p = \{p_w \colon w \in L\}$$

is recognizable.

Proof. Let (Q, q_0, δ, F) be a non-deterministic finite automaton recognizing the language. The idea of the proof is that we define a local language that accepts all valid runs of the automaton and then project the letters read during that run it to get rid of the states encoded in the language. The alphabet of the local language will consist of the valid transitions in the transition relation i.e each letter will know its previous and new state after the given letter has been read, and the tiling for the local language will ensure the consistency of the transitions between states. After this it only remains to check that the state we begun in was q_0 and the state we end in was one of the states in F; in the tiling this is done using the border symbol.

So recall that δ is a ternary relation between the source state, the letter read and the target state i.e $\delta \subseteq Q \times \Sigma \times Q$. Define alphabet $\Sigma' = \delta$. We then define our set of tiles T as the smallest set such that

- $((q', \alpha, q), (r, \alpha, r')) \in T$ for any $q', r' \in Q$ and $(q, \alpha, r) \in \delta$. This validates whether the current state was arrived in using a valid transition. Note that only one edge of each tile is considered—it is enough to verify that the tiles are "compatible" (think dominoes).

- $(\sharp, (q_0, \alpha, q)) \in T$ for any $\alpha \in \Sigma$ and $q \in Q$ such that $(q_0, \alpha, q) \in \delta$. This encodes the states that are reachable from the initial state which we can consider to be happening on the border.

- $((q, \alpha, r), \sharp) \in T$ for any $q \in Q$, $\alpha \in \Sigma$ and $q_f \in F$ such that $(q, \alpha, r) \in \delta$. The tiles at right border verify that the run was an accepting run i.e it ends in an accepting state.

Let L' be the local language defined by the tiling T and define language L as the projection of L' using projection $\pi \colon L' \to L$ defined as

$$\pi((r, \alpha, p)) = \alpha.$$

To show that $L = L_p$, it is, by corollary 1.1 above, enough to show given a non-empty word $w = \alpha_1 \dots \alpha_n$ we have $p_w \in L$ iff there exists a sequence

$$((q_0, \alpha_1, r_1), \dots, (r_{n-1}, \alpha_n, r_n)) \in \delta^n$$

such that $r_n \in F$. But given that this is essentially our definition of T above, it is equivalent to that for a picture p it holds that $T_{2,2}(p) \subseteq T$. Note that our border tiles on both ends ensure that exactly in this case we also have $p(1,1) = \pi((q_0, \alpha_1, q))$ for some $q \in Q$ and $p(1,n) = \pi((q, \alpha_n, r))$ for some $q \in Q$ and $r \in F$. □

Proposition 2.2. *Given a word-like recognizable language L_p over Σ, the word language $L_w = \{w \in \Sigma^\star \colon p_w \in L_p\}$ is recognized by a non-deterministic finite automaton.*

Proof. Let L'_p be the local language and $\pi \colon \Sigma' \to \Sigma$ the projection defining the recognizable language L_p. Let T be the set of allowed tiles of the local language L'_p.

Define the set of states Q and the set of final states F as

$$\begin{aligned} Q &= T \cup \{q_0\} \\ F &= T \times (T \cap (\Sigma' \times \{\sharp\})), \end{aligned}$$

where q_0 is a new state to be used as the initial state, and let the transition relation $\delta \subseteq Q \times \Sigma' \times Q$ be defined as the smallest relation such that

- $((a,b), \pi(b), (b,c)) \in \delta$ for $(a,b), (b,c) \in T$. This is used to recognize all possible tile-moves given a letter $\pi(b)$ of the projected alphabet. When the NFA performs this move, it means the letter read was a projection of a letter connecting two tiles, which ensures consistent tiling.
- $(q_0, \pi(a), (\sharp, a) \in \delta$ for any $(\sharp, a) \in T$. This recognizes valid initial letters in the tiling as the possible transitions from our initial state.

Define now non-deterministic finite automaton as $M = (Q, q_0, \delta, F)$. The idea of this automaton is that each non-initial state represents an allowed tile $(a,b) \in T$ with the idea that the last letter read was $\pi(b)$. The use of the tilings as states ensures that our tiling stays consistent between steps of the NFA even though we are reading the projected alphabet.

By corollary 1.1 above and our definition of δ, the following are equivalent given a word $w = \alpha_1 \dots \alpha_n$.

- The automaton M accepts word w.

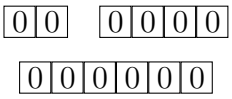
- There exists a sequence

$$((q_0, \pi(\alpha_1), (\sharp, a_1)), \ldots, ((a_{n-2}, a_{a-1}), \pi(\alpha_n), (a_{n-1}, a_n))) \in \delta^n$$

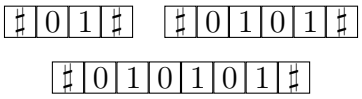
such that $(a_{n-1}, a_n) \in F$ i.e $(a_n, \sharp) \in T$. Since $\pi(\alpha_1 \ldots \alpha_n) = w$, $p_w \in L_p$.

□

Figure 6: Language L_{even} is the language of word-like pictures with even width.



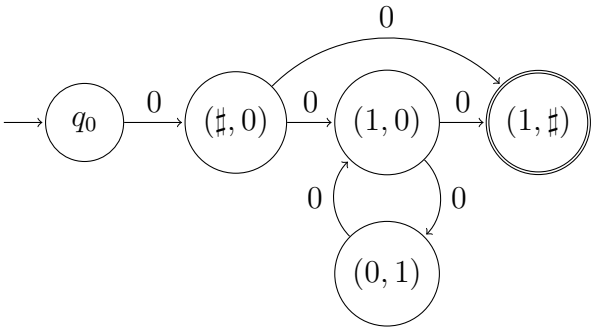
(a) some pictures of L_{even}



(b) corresponding bordered pre-images of L'_{even}

(c) The tile set T_{even} given as 1×2-tiles

Figure 7: The automaton constructed to recognize L_{even}.



Example 2.4. As an example of the above, we examine the recognizable (but not local) language L_{even} of word-like pictures over one letter alphabet $\{0\}$ defined as

$$L_{\text{even}} = \{p \in \{0\}^{\star\star} \colon l_1(p) = 1,\ l_2(p) \text{ is non-zero and even}\}.$$

Examples of pictures of L_{even} can be seen in figure 6a. We see that L_{even} is recognizable by projecting the local language L'_{even}—examples of which are shown in figure 6b—which in turn is local based on tiling T_{even} shown in

figure 6c. The tiling is given as 1×2-tiles for simplicity, see remark 2.1 for the justification and their interpretation to 2×2-tiles.

In figure 7 is shown the NFA as constructed in the proof of proposition 2.2 that accepts the corresponding word language of even words. Of course, a much simpler and deterministic automaton would do for this specific language.

2.3 Domino tilings

[LS97] introduced domino tilings that are also called hv-local, where hv stands for horizontal-vertical. In the definition above tilings were defined as squares, but using hv-local tilings we define allowed tiles as a set of 1x2 and 2x1 tiles. This has a computational interpretation—verifying hv-local tiling can be done separately for both horizontal and vertical directions whereas verifying a square tiling requires examining at least two rows during each horizontal scan if scans are performed row-wise.

Definition 2.9. A picture language L over Σ is *hv-local* if there exists a set Δ of pictures over Σ with size $(1, 2)$ or $(2, 1)$ such that for each $p \in L$ it holds that $T_{1,2}(p) \cup T_{2,1}(p) \subseteq \Delta$.

Note that the orientation of the dominoes allows us to keep track of direction in this definition. As an equivalent definition we could give distinct sets of two-symbol dominoes for each dimension as we will do in the third section.

The following two propositions show that hv-local languages are a proper subclass of recognizable languages.

Proposition 2.3. *A hv-local language L is also a local language.*

Proof. See [GR97]. □

Proposition 2.4. *There exists a language L that is local but not hv-local.*

Proof. See example in [GR97]. □

It turns out, however, that projections are quite powerful enough to make do with these tiles as the following theorem from [LS97] shows:

Theorem 2.4. *Given a language L the following are equivalent:*

- *L is recognizable*
- *L is a projection of a hv-local language.*

Proof. For a simple proof see [GR97]. Note that one direction is easy since hv-local language is a proper subclass of local languages, but for the other direction one needs to use the power of projections. □

The above theorem justifies our use of domino tilings as the definition of recognizability in the third section.

2.4 Automata on pictures

In this section we will consider generalizations of automata from word languages into picture languages, and we will see that the recognizability by a tiling system introduced in the previous section actually corresponds to one of them—non-deterministic online tessellation automata. We will consider most of the automata only informally and without proofs. A comprehensive survey is to be found in [GR97] and [KS11].

The challenge when generalizing one-dimensional automata to pictures is to decide how to scan the picture: in word languages this is trivial because there is just one way to walk through all the letters, and this is the reason why many generalizations with differing closure properties and expressive power become the same when restricted to one-dimensional case.

The automata on pictures can be divided, based on how the picture is scanned, to picture-walking automata, that literally walks the picture letter by letter, and to cellular automata like automata, that calculates state for all positions simultaneously. Both approaches result in same automata when restricted to one-dimensional case, but while the former tend to be too weak in the two-dimensional case, the latter, on the other hand, tend to become too powerful.

Four-way automata

Four-way automata introduced by [BH67] is possibly the most straightforward generalization of finite automata into pictures: the automaton has a state and can move in four directions instead of only ahead—it directly generalizes 2DFA and 2NFA from word languages.

While both deterministic four-way automata (4DFA) and non-deterministic four-way automata (4NFA) are closed under unions and intersections, and 4DFA additionally under complementation, they lack some closure properties such as closure under column and row concatenation operations. Whether 4NFA is closed under complementation is an open problem. Four-way automata belong to the larger family of picture-walking automata, and they are interesting in their own right. A recent survey on picture-walking automata can be found in [KS11].

Considering our above characterization of recognizable languages using hv-local tilings and our proof of proposition 2.2, one might think that we could construct a similar four-way automaton to first walk all rows and then all columns of a picture in the same fashion to see whether the projected tiling was allowed. The same technique, however, does not work in two-dimensional case: in our proof of the proposition 2.2, we examined not simply one tile at a time but two, to maintain consistency in our guesses for projected tiles. This works fine when there is only one direction to go and one needs only one tile worth of extra memory, but if we were to verify rows in this fashion first, we would not be able to remember our guessed tiles later when later verifying columns without using additional cell-specific memory. This is the reason why the definitions collapse when there is essentially one dimension only. The next theorem also shows that there is no hope of finding another way to do this by any clever means.

Theorem 2.5. *The class of languages accepted by 4NFA is properly included in* REC.

Proof. See [GR97]. □

On the other hand if we do have cell-specific memory, we can indeed do it, like the online tessellation automata introduced next shows us.

Online tessellation automata

Online tessellation automata was introduced by [IN77] and it also comes in both deterministic (DOTA) and non-deterministic (OTA) variants, and again—unlike automata for word languages—they are not equivalent.

The idea is simple: instead of walking through the picture cell-by-cell, the bordered picture of size (m, n) will be diagonally scanned so that at time t the state of every cell (i, j) such that $(i-1)+(j-1) = t-1$ will be determined by the transition relation given the states already calculated for preceeding cells $(i-1, j)$ and $(i, j-1)$. The state of the cell at bottom-right corner of the picture determined at time $t = \max(m, n)$ decides whether the run was an accepting one or not. Otherwise the definition is much like that of a usual word automaton.

Definition 2.10 (Online tessellation automata)**.** Given an alphabet Σ, a *non-deterministic two-dimensional online tessellation automaton (OTA)* is defined by a tuple (Q, q_0, F, δ) where

- Q is a finite set of states

Figure 8: Diagonal scan by an online tessellation automaton

♯	♯	♯	♯	♯
♯	$(1,1)$	$(0,2)$	$\cdots$	♯
♯	$(2,0)$	$\ddots$	$(i-1,j)$	♯
♯	$\cdots$	$(i,j-1)$	(i,j)	♯
♯	♯	♯	♯	♯

- q_0 is the initial state (for border)
- F is the set of accepting states
- $\delta \subseteq Q \times Q \times \Sigma \times Q$ is the transition relation.

The automaton is *deterministic* (DOTA) if δ is a function $Q \times Q \times \Sigma \to Q$.

It was shown in [IN77] that the deterministic version is strictly less powerful.

Theorem 2.6. *Languages recognized by DOTA are a strict subset of those recognized by OTA.*

Non-deterministic online tessellation automata turn out to be equivalent to the recognizable languages (characterized by tiling systems) introduced in the previous section as shown in [IT92].

Theorem 2.7. *The following are equivalent:*

- *L is recognized by an OTA*
- *L is recognizable.*

The proof of above theorem is not much different from our proof above that recognizable word-like picture languages correspond to regular word languages—if anything, it is simpler. The diagonal scan is well-suited to verify a square tiling and non-determinism now provides the projection. Note that the non-determinism present here is more potent than is possible for non-deterministic word automata or non-deterministic four-way automata—the difference intuitively being that an OTA can guess all cells at once and then verify the tiling, whereas 4NFA can only guess one cell at a time and cannot remember its past guesses for more than a bounded number of cells.

3 Logical locality on picture languages

In [GRST96] it was shown that the class of two-dimensional recognizable picture languages coincides with the class of languages defined by EMSO. The paper finds an interesting characterization of first-order logic on pictures using tilings and then introduces projections of recognizable languages which correspond to predicate quantification of EMSO. The classical tools of finite model theory such as back-and-forth games and partial isomorphisms are used in the proof. In [GO16] the characterization was improved by extending the result to any dimension and by way of a different proof presenting an alternative perspective that uses syntactical normalization of logic to a sublogic with strong locality properties among those structures. It is this latter proof we present here. Except when otherwise mentioned, all the proofs in this section are based on [GO16].

In this section we will also redefine pictures and tilings for any dimension but as squares i.e having the same side length for all dimensions. The reason is to simplify the notation. It can be seen that the proofs in this section do not actually depend on the fact that the pictures are squares as each dimension is always handled rather independently, so the results could be generalized to pictures of any shape. Our use of domino tilings in the definition here instead of square tilings is justified by the theorem 2.4 above.

It will be seen that a recurring theme in the proofs is the ability to define unary predicates inductively both only in one dimension and also on all elements of a picture using lexicographic ordering, which is easily expressible in pixel structures even without quantifying any extra predicate symbols. We will proceed by first showing that EMSO on pixel structures is in fact equivalent to syntactically much more limited forms of EMSO, first to EMSO($\forall^1$) which allows only one universal quantifier in the first-order part and then to a further normalized form the syntax of which resembles tilings, which in turn makes it easy to show that REC and EMSO define the same picture languages. Throughout the proofs in this section, it will be useful to keep in mind that in EMSO unary predicates in the signature correspond to the alphabet and quantified unary predicates to the guessed symbols, which in REC are represented by the projection.

Definition 3.1 (in this section only)**.** A *d-dimensional picture* is any function $p\colon [n]^d \to \Sigma$. Given a picture p we write $\mathrm{dom}(p)$ for $[n]^d$ and call it the *domain* of the picture. Elements of $\mathrm{dom}(p)$ are called *points* or *pixels* of p.

Definition 3.2 (in this section only)**.** Given alphabet Σ, we write $\Sigma^\sharp$ for alphabet $\Sigma \cup \{\sharp\}$ where $\sharp$ is a new special symbol called the *border symbol*

that is not in Σ. Given a d-picture $p\colon [n]^d \to \Sigma$, we write $p^\sharp\colon [n+2]^d \to \Sigma$ for the *bordered picture* of p (over $\Sigma^\sharp$) defined as:

$$p^\sharp(a_1, \ldots, a_d) = \sharp, \text{ when any of } a_1, \ldots, a_d \text{ is } 1 \text{ or } n+2$$
$$p^\sharp(a_1, \ldots, a_d) = p(a_1 - 1, \ldots, a_d - 1), \text{ otherwise.}$$

Next, we define the *pixel structures* for examining model theoretic and logical properties of pictures.

Definition 3.3. Given a picture $p\colon [n]^d \to \Sigma$, we define $\mathrm{pixel}^d(p)$ as structure

$$\left([n]^d, (Q_s)_{s\in\Sigma}, (\mathrm{succ}_i)_{i\in[d]}, (\min_i)_{i\in[d]}, (\max_i)_{i\in[d]}\right)$$

where

- succ_i is the cyclic successor function in the ith component:

  $$\mathrm{succ}_i(a_1, \ldots, a_d) = (a_1, \ldots, a_i + 1, \ldots, a_n)$$

  when $a_i < n$ and

  $$\mathrm{succ}_i(a_1, \ldots, a_d) = (a_1, \ldots, 1, \ldots, a_n)$$

  when $a_i = n$.

- $\min_i$, $\max_i$ and Q_s are the unary predicates

  $$\min_i = \{a \in [n]^d \colon a_i = 1\}$$
  $$\max_i = \{a \in [n]^d \colon a_i = n\}$$
  $$Q_s = \{a \in [n]^d \colon p(a) = s\}$$

The use of successor functions instead of relations is important here for examining locality: if we use ordering relation instead, every pixel is directly connected to every other pixel. The logical characterization itself would still be the same. Also note that the cyclicity of the successor functions is only needed to make them proper permutations of the domain which simplifies the proofs. The following remark will be evident from the normal form introduced in the proposition 3.4 later.

Remark 3.1. Any formula of EMSO can be written so that its interpretation on pixel structures is the same regardless of how the pixels "on the edges" (having any coordinate of n) are interpreted.

This is due to the fact that in the normalized form presented in proposition 3.4, only terms of form x and $\mathrm{succ}_i(x)$ occur and the latter form only ever occurs under the premise that $\neg \max_i(x)$ holds for x. Thus the interpretation of $\mathrm{succ}_i(x)$ in the model at pixel x such that $\max_i(x)$ cannot change the interpretation of the sentence.

3.1 Elimination of quantifiers in bijective structures

We will first introduce a quantifier elimination result adapted from [DG07] that we will then use to normalize our logic on pixel structures.

Definition 3.4. We call signature $\sigma = (U_1, \ldots, U_k, f_1, \ldots, f_d, f_1^{-1}, \ldots, f_d^{-1})$ *monadic* if each U_i is a unary predicate and f_i and f_i^{-1} are unary function symbols. A structure with a monadic signature is called *bijective* if the interpretation of each f_i is a permutation of the structure's domain and the interpretations of f_i^{-1} and f_i for each i are bijective inverses of each other.

We define $\exists^{\geq k}$ as shorthand

$$\exists^{\geq k} x\, \phi(x) = \exists x_1 \ldots \exists x_k \left(\bigwedge_{1 \leq i < j \leq k} \neg x_i = x_j \bigwedge_{1 \leq i \leq k} \phi(x_i) \right).$$

We assume the transformation to be performed so that all used variables are new and in the following when e.g renaming variables, all necessary renamings are done in the obvious way.

Definition 3.5. A *cardinality formula* is a first-order formula of the form $\exists^{\geq k} x\, \phi(x)$ where k is a positive integer and $\phi(x)$ is a quantifier-free formula with x as its only free variable.

So intuitively a cardinality formula is a sentence that asserts there to be at least k elements in the structure with a specified local property, locality here meaning anything expressible without quantifiers.

Proposition 3.1. *Given a signature $\sigma = (U_1, \ldots, U_k, f_1, \ldots, f_d, f_1^{-1}, \ldots, f_d^{-1})$, every first-order formula is equivalent to a boolean combination of atomic formulae and cardinality formulae on bijective structures.*

Proof. We will prove the proposition by induction on the construction of first-order formulae. Note that as our target is a boolean combination, the only construct with anything to prove will be existential quantification.

Assume then that $\phi(y, \bar{x})$ is a boolean combination of atomic and cardinality formulae. We need to show that $\exists y\, \phi(y, \bar{x})$ can be transformed into the required form. We will first simplify our case further by noting the following:

1. We can assume $\phi(\bar{x})$ to be in a disjunctive normal form—considering existing cardinality formulae atomic for the purposes of this transformation.

2. Existential quantification commutes with disjunction, so we may deal with disjuncts separately and assume $\phi(\bar{x})$ to be a conjunction of (possibly negated) atomic formulae and cardinality formulae.

3. Conjuncts that do not have y as a free variable can be dealt with by simply removing the quantifier. Especially this includes any (possibly negated) cardinality formulae as they have no free variables by definition. Thus we can assume $\phi(\bar{x})$ to be a conjunction of literals. Furthermore, as each literal not using identity can only involve one variable due to our monadic vocabulary, we can assume all non-identity literals have only y as their free variable.

4. Due to our bijectivity assumption, each term is invertible and we may assume each identity atomic formula to be written in form $y = \tau(x_i)$ where $\tau(x_i)$ is a term and $x_i \in \bar{x}$.

5. If any conjunct is of form $y = \tau(x_i)$, we are done, as we can replace y with $\tau(x_i)$ in the conjunction and remove the quantification as it no longer involves variable y. So we may assume every atomic formula using identity is a negative literal.

Given the above considerations, we may assume that $\phi(\bar{x})$ is of form

$$\psi(y) \wedge y \neq \tau_1(x_1) \wedge \cdots \wedge y \neq \tau_k(x_k),$$

where $\psi(y)$ is a conjunction of literals that make no use of identity atoms and $x_i \in \bar{x}$ for $1 \leq i \leq k$. Note that we do not require each x_i to be different variables, we just want to simplify the notation.

It is good to pause and observe that the following holds trivially:

$$\exists^{k+1} y\, \psi(y) \Rightarrow \exists y\, (\psi(y) \wedge y \neq \tau_1(x_1) \wedge \cdots \wedge y \neq \tau_k(x_k)).$$

The problem for the converse is that some of the terms $\tau_1(x_1), \ldots \tau_k(x_k)$ might get the same interpretation or might fail to satisfy $\psi(\tau_i(x_i))$, in which case we could satisfy the original formula with less than $k+1$ entities.

Given a structure $\mathcal{S}$, let $P \subseteq [k]$ be the indices of terms the interpretation of belongs to $\psi(S)$ i.e $\psi(\tau_i(x_i))$ holds for $i \in P$ and $\neg\psi(\tau_i(x_i))$ holds for $i \in [k] \setminus P$. Let $Q \subseteq P$ be the set of indices such that for any $i \in P$ there is $j \in Q$ such that $\tau_i(x_i) = \tau_j(x_j)$ holds in the structure, and let $h = |Q|$. Now h is an upper bound for number of different interpretations for terms $\tau_i(x_i)$ that satisfy $\psi(\tau_i(x_i))$, and if h is maximal, we will have

$$\exists^{h+1} y\, \psi(y) \Leftrightarrow \exists y\, (\psi(y) \wedge y \neq \tau_1(x_1) \wedge \cdots \wedge y \neq \tau_k(x_k))$$

in the structure.

Luckily, we can encode the above conditions easily syntactically, and then we can simply try all possible combinations.

$$\phi'(\bar{x}) = \bigvee_{h=0}^{k} \bigvee_{\substack{P \subseteq [k] \\ Q \subseteq P, |Q|=h}} \begin{bmatrix} \bigwedge_{j \in Q} \psi(\tau_j(x_j)) \wedge \\ \bigwedge_{i \in P} \bigwedge_{j \in Q} \tau_i(x_i) = \tau_j(x_j) \wedge \\ \bigwedge_{j \in [k] \setminus P} \neg\psi(\tau_j(x_j)) \wedge \\ \exists^{h+1} y\, \psi(y) \end{bmatrix}$$

Now in any structure $\mathcal{S}$ it holds that

$$\phi'(\bar{x}) \Leftrightarrow \exists y\, (\psi(y) \wedge y \neq \tau_1(x_1) \wedge \cdots \wedge y \neq \tau_k(x_k))$$

□

Corollary 3.1. *Given a signature σ as above, every first-order sentence is equivalent to a boolean combination of cardinality formulae on bijective structures.*

Proof. Result follows from above proposition immediately: the obtained formula will have the same free variables and will thus be an equivalent sentence. Note that an atomic formula can only occur inside a cardinality formula as otherwise it would introduce a free variable. □

This result has a computational interpretation: any first-order property of a bijective structure can now be evaluated by taking a boolean combination of properties that in turn can be checked by going through elements of the model and evaluating a local property for each and only doing bounded counting during the scan (namely up to the cardinality of the cardinality formulae used). In other words this means that for a fixed formula the expressed property is checked in linear time and constant space.

3.2 Normalizing logic on pixel structures

We will now use the elimination of quantifiers from the previous section to show that anything expressible in EMSO on pixel structures can be expressed by a sentence using only a single universal first-order quantification. We first apply the result from the previous section to pixel structures:

Corollary 3.2. *On pixel structures, each first-order sentence is equivalent to a boolean combination of cardinality formulae.*

Proof. If we extend our pixel structures with corresponding pred_i function for each succ_i and $i \in [d]$, corollary 3.1 from the previous section clearly applies as the structures become bijective with this signature. If we can now show that each pred_i can be expressed using only succ_i given that the sentence is a boolean combination of cardinality formulae, we can eliminate the new symbols and thus establish the result for our original signature.

But certainly a cardinality formula $\exists^{\geq k} x\, \phi(x)$ is equivalent to $\exists^{\geq k} x\, \phi(\mathrm{succ}_i(x))$ for any $i \in [d]$, as each succ_i is bijective. Furthermore, all the successor and predecessor functions commute with each other so we can simply use above observation, substitute $\mathrm{succ}_i(x)$ for x for every pred_i mentioned in the formula and then eliminate the corresponding pairs of them using commutativity. Thus all pred_i functions have been eliminated and we are left with a boolean combination of cardinality formulae in the original signature. □

Of course, the above result applies to any class of bijective structures where the interpretations of the functions always commute with each other. Next we use the inductive power of pixel structures and bounded counting ability of EMSO to eliminate the cardinality formulae.

Proposition 3.2. $\mathrm{EMSO} \subseteq \mathrm{EMSO}(\forall^1)$ *on pixel structures.*

Proof. Let $\exists \bar{U} \phi$ be a sentence of EMSO. By the previous corollary, we can assume the first-order part ϕ to be a boolean combination of cardinality formulae. It will be enough to show that any cardinality formula of form $\exists^{\geq k} x\, \psi(x)$ or $\neg \exists^{\geq k} x\, \psi(x)$ can be expressed as a sentence of $\mathrm{EMSO}(\forall^1)$ as we can assume the boolean combination of cardinality formulae to be in disjunctive normal form and $\mathrm{EMSO}(\forall^1)$ is closed under conjunction and disjunction.

The idea is to use $\min_i$, $\max_i$ and succ_i functions to iterate the structure in lexicographic order and to use k new relation symbols $(U_j)_{1 \leq j \leq k}$ to perform bounded counting up to k: the intended meaning of $U_j(x)$ being that there are at least j pixels satisfying ψ in the structure up to x in the lexicographic order.

Let us temporary expand our signature with $\min_{\mathrm{lex}}$, $\max_{\mathrm{lex}}$ and $\mathrm{succ}_{\mathrm{lex}}$ interpreted as lexicographic order on $[n]^d$ given a pixel structure with domain $[n]$. Given a quantifier-free formulae $\psi(x)$, define $\psi'(x)$ as conjunction of the following formulae

$$\min_{\mathrm{lex}}(x) \to \left[(\psi(x) \leftrightarrow U_1(x)) \wedge \bigwedge_{1 < i \leq k} \neg U_i(x) \right]$$

$$\neg \max_{\mathrm{lex}}(x) \to \bigwedge_{1 \leq i \leq k} \left[\left(U_i(x) \vee \left[\psi(\mathrm{succ}_{\mathrm{lex}}(x)) \wedge \bigwedge_{1 \leq j < i} U_j(x) \right] \right) \leftrightarrow U_i(\mathrm{succ}_{\mathrm{lex}}(x)) \right].$$

Now given any pixel structure, there is always a unique interpretation for $U_1, \ldots, U_k$ satisfying $\forall x\, \psi'(x)$. Furthermore, this makes $\exists^{\geq k} x\, \psi(x)$ equivalent to

$$\exists U_1 \ldots U_k \forall x\, [\psi'(x) \wedge \max_{\text{lex}}(x) \rightarrow U_k(x)]$$

and $\neg\exists^{\geq k} x\, \psi(x)$ to

$$\exists U_1 \ldots U_k \forall x\, [\psi'(x) \wedge \max_{\text{lex}}(x) \rightarrow \neg U_k(x)]$$

Now it only remains to get rid of $\min_{\text{lex}}$, $\max_{\text{lex}}$ and succ_{lex} that are not part of our our vocabulary. This must be done while observing the requirement for the final formula.

Note first, that $\text{succ}_{\text{lex}}(x) = \text{succ}_i \ldots \text{succ}_d(x)$ for smallest $i \in [d]$ such that $\max_j(x)$ holds for all $j > i$. Suppose $\phi(\text{succ}_{\text{lex}}(x))$ is a formula involving $\text{succ}_{\text{lex}}(x)$ term. We may rewrite it equivalently as

$$\bigwedge_{i \in [d]} \left(\left[\neg \max_i(x) \wedge \bigwedge_{i < j \leq d} \max_j(x) \right] \rightarrow \psi(\text{succ}_i \ldots \text{succ}_d(x)) \right).$$

To rewrite $\psi(\max_{\text{lex}}(x))$ and $\psi(\min_{\text{lex}}(x))$ we simply write

$$\bigwedge_{i \in [d]} \max_i(x) \rightarrow \psi(x)$$

and

$$\bigwedge_{i \in [d]} \min_i(x) \rightarrow \psi(x),$$

respectively. □

Reduction of EMSO to EMSO($\forall^1$) is already quite a strong normal form when locality is considered: it essentially states that a bunch of properties (existential monadic second-order quantification) holds uniformly in the structure (universal first-order quantification) where each property is determined for each pixel by a bounded local neighbourhood (only successor functions used to refer to other elements). Indeed, in the above proof more was seen: cardinality statements of EMSO were converted into a sort of a "scan" through the structure. In the next section, we will use similar ideas and reduce EMSO($\forall^1$) even further.

3.3 Tiling pictures

As seen in section 2 above, tilings are an interesting way to look at picture languages due to their inherent locality properties. In this section, we will

use tilings to define the class of recognizable d-dimensional picture languages REC^d and show that it in fact coincides with $\mathrm{EMSO}(\forall^1)$.

We will begin by introducing two normalizations of $\mathrm{EMSO}(\forall^1)$. The first is to observe that identity statements are not required to talk about pixel structures. This makes sense intuitively—as we are only quantifying pixels using one universal quantifier and we can only use terms that include successor functions that are permutations, any identity atomic formulae would in fact evaluate the same for all pixels and simply express a statement about the size of the picture.

Proposition 3.3. *On pixel structures, a sentence of* $\mathrm{EMSO}(\forall^1)$ *is equivalent to an* $\mathrm{EMSO}(\forall^1)$ *sentence that contains no identity atoms i.e atomic formulae of form* $\tau_1(x) = \tau_2(x)$.

Proof. Let $\exists \bar{U} \forall x \, \psi(x)$ be a sentence of $\mathrm{EMSO}(\forall^1)$. Note that ψ is quantifier-free. Also assume ψ to be in negative normal form meaning that negation symbols occur only in front of atomic formulae.

Consider first what equalities mean on pixel structures. An atomic formula $\mathrm{succ}_i^k(x) = x$ is true in the pixel structure of a picture p with domain $[n]$ iff k is a multiple of n. Also note that due to the nature of successor functions in our square pictures, $\tau_1(x) = \tau_2(x)$ holds for one pixel iff it holds for all pixels. Given an identity statement for $c > 0$ and τ_1 and τ_2 such that they do not contain succ_i, it holds on pixel structures that

$$\mathrm{succ}_i^c(\tau_1(x)) = \tau_2(x) \leftrightarrow (\mathrm{succ}_i^c(x) = x \wedge \tau_1(x) = \tau_2(x)) \, .$$

This follows from two facts: that due to the nature of successor functions any identity statement either holds for all pixels or none, and that distinct successor functions are independent. Thus we may assume without loss of generality that each identity atomic formula is of form $\mathrm{succ}_i^c(x) = x$ for some $i \in [d]$ and $c > 0$.

Let $\mathrm{succ}_i^c(x) = x$ then be an atomic identity formula of $\psi(x)$. The idea is simple: we use bounded counting as we did in the last section—only simpler—to count successors from a $\min_i$ pixel to $\max_i$ pixel. $\mathrm{succ}_i^c(x) = x$ holds iff we count exactly c successors between the pixels. For each $k \in [c]$ we introduce a new symbol $C_i^{=k}$ and let

$$\delta_i^c(x) = \left(\min_i(x) \leftrightarrow C_i^{=1}\right) \wedge \bigwedge_{j \in [c-1]} \left(C_i^{=j}(x) \leftrightarrow C_i^{=j+1}(\mathrm{succ}_i(x))\right) .$$

Note that when the above holds in a pixel structure, $C_i^{=c}$ holds of the pixel having $\max_i$ exactly when k is a multiple of n since we begin counting at $\min_i$ and let it cycle to min if needed.

Now $\exists \bar{U} \forall x \, \psi(x)$ is equivalent to

$$\exists \bar{U} \exists_{j \in [c]} C_i^{=j} \forall x \, (\delta_i^c(x) \wedge \psi'(x)) \, ,$$

where $\psi'(x)$ has been obtained by substituting every positive literal $\mathrm{succ}_i^c(x)$ with

$$\max_i(x) \to C_i^{=c}(x).$$

and every negative literal $\neg \, \mathrm{succ}_i^c(x)$ with

$$\max_i(x) \to \neg C_i^{=k}(x).$$

We can now repeat this replacement until no more identity statements remain in the sentence. □

Next we use similar techniques to remove term nesting, normalize "direction" of successors and make the overall form more inductive, removing cyclicity. Note that this logical form is essentially an inductive verification of a guessed (cf. projection) hv-local tiling (see section 2.3) for each dimension separately.

Proposition 3.4. *On pixel structures, a sentence of* $\mathrm{EMSO}(\forall^1)$ *is equivalent to a sentence of form*

$$\exists \bar{U} \forall x \bigwedge_{i \in [d]} \begin{bmatrix} \min_i(x) \to m_i(x) \wedge \\ \max_i(x) \to M_i(x) \wedge \\ \neg \max_i(x) \to \psi_i(x) \end{bmatrix},$$

where m_i, M_i *and* ψ_i *are quantifier-free formulae such that atomic formulae of*

- $m_i(x)$ *and* $M_i(x)$ *are of form* $Q(x)$
- $\psi_i(x)$ *are of form* $Q(x)$ *or* $Q(succ_i(x))$,

with each $Q \in \{(Q_s)_{s \in \Sigma}, \bar{U}\}$.

Proof. Let now $\exists \bar{U} \forall x \, \psi(x)$ be a sentence of $\mathrm{EMSO}(\forall^1)$. In the light of previous proposition, we may assume that $\psi(x)$ contains no identity atomic formulae.

For a unary predicate symbol Q, define for each term τ a new relation symbol $U_{Q,\tau(x)}$ and then by induction on construction of terms the corresponding formulae $\psi_{Q,\tau(x)}$ as follows:

$$\psi_{Q,x} = U_{Q,x} \leftrightarrow Q(x)$$

$$\psi_{Q,\mathrm{succ}_i(\tau(x))} = \Big(U_{Q,\mathrm{succ}_i(\tau(x))} \leftrightarrow U_{Q,\tau(x)}(\mathrm{succ}_i(x))\Big) \wedge \psi_{Q,\tau(x)}.$$

The idea here is to remove nesting in terms by guessing a predicate that knows how the term will be evaluated for the predicate we are interested in.

Let $\psi'(x)$ be the formula resulting from simultaneously replacing each atomic formula $Q(\tau(x))$ of $\psi(x)$ with $U_{Q,\tau(x)}(x)$. Now certainly $\exists \bar{U} \forall x\, \psi(x)$ is equivalent to

$$\exists \bar{U} \exists_{Q(\tau(x))\in\psi} (U_{Q,\tau(x)}) \forall x \left(\psi'(x) \wedge \bigwedge_{Q(\tau(x))\in\psi} \psi_{Q,\tau(x)} \right),$$

where $Q(\tau(x)) \in \psi$ means that $Q(\tau(x))$ is an atomic formula of ψ.

We introduce some terminology to help complete the proof. By a *clause* we mean a disjunction of literals. Note that a clause $C_1 \vee \cdots \vee C_n$ may be also be written equivalently as $\neg C_1 \to (C_2 \vee \cdots \vee C_n)$. A clause is

- *pure* if it only contains atoms of the form $Q(x)$ where Q is not $\min_j$ nor $\max_j$ for any $j \in [d]$.
- *i-cyclic* if it only contains atoms of the form $Q(x)$ or $Q(\mathrm{succ}_i(x))$ where Q is not $\min_j$ nor $\max_j$ for any $j \in [d]$.
- *i-local* if it is of a form $\neg \max_i(x) \to C(x)$ where $C(x)$ is an i -cyclic clause.
- *i-min* if if it is of the form $\min_i(x) \to C(x)$ where $C(x)$ is a pure clause.
- *i-max* if if it is of the form $\max_i(x) \to C(x)$ where $C(x)$ is a pure clause.

Note that the claim of the proposition is now equivalent to showing that

$$\psi'(x) \wedge \bigwedge_{Q(\tau(x))\in\psi} \psi_{Q,\tau(x)}$$

is equivalent to a conjunction of clauses such that each clause is either pure, i-min, i-max or i-local for some $i \in [d]$. Pure clauses are not allowed in the requirement, but note that a pure clause $C(x)$ of the conjunctive normal form is equivalent to

$$\bigwedge_{i\in[d]} \begin{pmatrix} \min_i(x) \to C(x) \wedge \\ \max_i(x) \to C(x) \wedge \\ \neg \min_i(x) \to C(x) \end{pmatrix}.$$

Put $\psi'(x)$ in conjunctive normal form. Note that each clause of $\psi'(x)$ is pure, because it only makes use of term x along with relation symbols $Q_{\tau(x)}$ for any relation symbol Q and term $\tau(x)$ mentioned in ψ.

So we are left to deal with conjuncts of form $\psi_{Q,\tau(x)}$ for each $Q(\tau(x))$ mentioned in the original formula ψ. We will handle $\psi_{Q,x}$ and $\psi_{Q,\mathrm{succ}_i(\tau'(x))}$ separately.

For $\psi_{Q,x}$ we have tree cases:

- If Q is not $\min_i$ nor $\max_i$ for any $i \in [d]$, $\psi_{Q,x}$ is pure and can be handled as above.
- If $Q = \min_i$ for $i \in [d]$, the definition of $\psi_{Q,x}$ becomes
$$U_{Q,x}(x) \leftrightarrow \min_i(x)$$
which is equivalent to
$$(\min_i(x) \to U_{\min_i,x}(x)) \wedge (\neg \min_i(x) \to \neg U_{\min_i,x}(x)) .$$
The first conjunct here is i-min already. The second conjunct can be replaced with i-local clause
$$\neg \max_i(x) \to \neg U_{\min_i,x}(\mathrm{succ}_i(x)),$$
which is equivalent due to our universal quantification.
- If $Q = \max_i$, $\psi_{\max_i,x}$ is equivalent to
$$(\max_i(x) \to U_{\max_i,x}(x)) \wedge (\neg \max_i(x) \to \neg U_{\max_i,x}(x)) .$$
Here already the first conjunct is i-max and the second one i-local so there is nothing more to do.

Let a term $\tau(x)$ then be given. Formula $\psi_{Q,\mathrm{succ}_i(\tau(x))}$ is defined inductively as a conjunction so we can assume $\psi_{Q,\tau(x)}$ handled already ($\psi_{Q,x}$ was handled above) and deal with the new conjunct introduced in the inductive step:
$$U_{Q,\mathrm{succ}_i(\tau(x))}(x) \leftrightarrow U_{Q,\tau(x)}(\mathrm{succ}_i(x)).$$
The i-local part is easily recognized, but we need to get rid of cyclicity on the $\max_i(x)$ pixel, when $\mathrm{succ}_i(x)$ refers to $\min_i$ pixel. This, however, is easily handled by another new unary relation symbol $U^{\min_i}_{Q,\tau(x)}$ that remembers $U_{Q,\tau(x)}(x)$ for $\min_i(x)$. Thus assuming quantification of the new relation symbol, we can replace $\psi_{Q,\mathrm{succ}_i(\tau(x))}$ with equivalent
$$\begin{aligned}
\min_i(x) &\to \left(U^{\min_i}_{Q,\tau(x)}(x) \leftrightarrow U_{Q,\tau(x)}(x)\right) \wedge \\
\neg \max_i(x) &\to \left(U^{\min_i}_{Q,\tau(x)}(x) \leftrightarrow U^{\min_i}_{Q,\tau(x)}(\mathrm{succ}_i(x))\right) \wedge \\
\neg \max_i(x) &\to \left(U_{Q,\mathrm{succ}_i(\tau(x))}(x) \leftrightarrow U_{Q,\tau(x)}(\mathrm{succ}_i(x))\right) \wedge \\
\max_i(x) &\to \left(U_{Q,\mathrm{succ}_i(\tau(x))}(x) \leftrightarrow U^{\min_i}_{Q,\tau(x)}(x)\right)
\end{aligned}$$
made up of i-max and i-local clauses. □

We now define a tiling system to classify pictures. We will make use of the bordering defined in the previous section. We first define notion of tilings and local picture languages.

Definition 3.6.

1. Given a d-picture p and $j \in [d]$, pixels $(a_1, \ldots, a_d)$ and $(b_1, \ldots, b_d)$ are said to be *j-adjacent* if $|a_j - b_j| = 1$ and $a_k = b_k$ for any $k \in [d]$ with $k \neq j$.

2. A *tile* for a d-language Σ is a pair in $(\Sigma^\sharp)^2$.

3. A picture $p\colon [d]^d \to \Sigma$ is *j-tiled* by a set of tiles $\Delta \subseteq (\Sigma^\sharp)^2$ if for any two j-adjacent $a, b \in \mathrm{dom}(p^\sharp)$ it holds that $(p^\sharp(a), p^\sharp(b)) \in \Delta$.

4. A d-picture p is *tiled* by $(\Delta_1, \ldots, \Delta_d)$, if p is j-tiled by Δ_j for each $j \in [d]$.

5. We write $\mathcal{L}(\Delta_1, \ldots, \Delta_d)$ for the set of d-pictures tiled by $(\Delta_1, \ldots, \Delta_d)$.

6. A d-language L on Σ is *$(\Delta_1, \ldots, \Delta_d)$-tiled* if $L = \mathcal{L}(\Delta_1, \ldots, \Delta_d)$.

7. A d-language L is *local* if it is $(\Delta_1, \ldots, \Delta_d)$-tiled by some sets of tiles $(\Delta_1, \ldots, \Delta_d)$.

Even though local picture languages is a rather weak notion, it becomes powerful once we add ability to use extra markings during recognition. This will be seen to correspond to the existential quantification of unary predicates in EMSO.

Definition 3.7. A d-language L on Σ is *recognizable* if it is the projection of a local d-language over alphabet Σ'. In other words, there exists a function $\pi\colon \Sigma' \to \Sigma$ and a local d-language L' such that

$$L = \pi(L') = \{\pi \circ p\colon p \in L'\}.$$

Note that $\pi \circ p$ is a d-picture over Σ.

The class of recognizable d-languages over Σ is denoted by $\mathrm{REC}^d(\Sigma)$ or simply REC^d when the alphabet used is understood.

Theorem 3.1. *For any $d > 0$ and a d-language L on Σ: $L \in \mathrm{REC}^d$ iff $\mathrm{pixel}^d(L) \in \mathrm{EMSO}(\forall^1)$.*

Proof. Assume first, that L is a recognizable d-language over alphabet Σ. Let $\Delta_1, \ldots, \Delta_d$ be sets of tiles over an alphabet Σ' and $\pi\colon \Sigma' \to \Sigma$ be such that $L = \pi(\mathcal{L}(\Delta_1, \ldots, \Delta_d))$. We may assume that alphabets Σ and Σ' are distinct.

First we validate the underlying local language. Define formula $\Psi_{\Delta_1,\ldots,\Delta_d}$ as

$$\Psi_{\Delta_1,\ldots,\Delta_d} = \bigwedge_{i\in[d]} \begin{bmatrix} \min_i(x) \to \bigvee\limits_{(\sharp,s)\in\Delta_i} Q_s(x) \wedge \\ \neg \max_i(x) \to \bigvee\limits_{(s,s')\in\Delta_i} (Q_s(x) \wedge Q_{s'}(\mathrm{succ}_i(x))) \wedge \\ \max_i(x) \to \bigvee\limits_{(s,\sharp)\in\Delta_i} Q_s(x) \wedge \end{bmatrix}.$$

Note that for a d-picture over Σ' it holds that

$$\mathrm{pixel}^d(p') \models \Psi_{\Delta_1,\ldots,\Delta_d} \text{ iff } p' \in \mathcal{L}(\Delta_1, \ldots, \Delta_d).$$

Now we only need to quantify symbols $(Q_s)_{s\in\Sigma'}$ and express that they respect projection π, and we can recognize L. That is, let Ψ_L be the EMSO($\forall^1$)-sentence

$$(\exists Q_s)_{s\in\Sigma'} \forall x\colon \Psi_{\Delta_1,\ldots,\Delta_d} \wedge \bigwedge_{s\in\Sigma} \left(Q_s(x) \to \left[\bigoplus_{s'\in\pi^{-1}(s)} Q_{s'}(x) \wedge \bigwedge_{s'\in\Sigma'\setminus\pi^{-1}(s)} \neg Q_{s'}(x) \right] \right),$$

where $\oplus$ means exclusive disjunction. The sentence expresses that pixel labelled Q_s must be labelled by exactly one of its inverse elements in π and by no other labels. We now have

$$p \in L \text{ iff } \mathrm{pixel}^d(p) \models \Psi_L.$$

Assume then, that pixel(L) is definable in EMSO($\forall^1$). By proposition 3.4 we may assume it to be defined by a sentence

$$\Psi_L = \exists \bar{U} \forall x \bigwedge_{i\in[d]} \begin{bmatrix} \min_i(x) \to m_i(x) \wedge \\ \max_i(x) \to M_i(x) \wedge \\ \neg \max_i(x) \to \psi_i(x) \end{bmatrix},$$

where m_i, M_i and ψ_i are as in proposition 3.4.

Let $\bar{U} = U_1, \ldots, U_k$. Put each m_i, M_i and ψ_i in complete disjunctive normal form i.e disjunctive normal form so that every clause in m_i, M_i and ψ_i mentions every atomic sentence of form $Q_s(x)$ and $U_j(x)$ for every $j \in [k]$, and ψ_i further every atomic sentence of form $Q_s(\mathrm{succ}(x))$ or $U_j(\mathrm{succ}_i(x))$—this is done by simply splitting each disjunct for every atomic formula not mentioned

in it until they become explicit. Also note that exactly one of $(Q_s(x))_{s\in\Sigma}$ and one of $(Q_s(\mathrm{succ}_i(x)))_{s\in\Sigma}$ appears as a positive literal in each clause of m_i, M_i and ψ_i. A clause having multiple positive occurrences of either could not be satisfied by any pixel structure. We will use this fact next.

Define new alphabet $\Sigma' = \Sigma \times \mathcal{P}([k])$ i.e attaching a subset of $[k]$ for each symbol of the original alphabet. We will use the symbols of Σ' to encode which of the unary predicates U_j hold for a given pixel. For each $i \in [d]$ let $\Delta_i \subseteq \left(\Sigma'^{\sharp}\right)^2$ be the union of the following sets:

- $((s,K),(s,K')) \in (\Sigma')^2$ such that ψ_i has a clause having exactly $Q_s(x)$, $Q_{s'}(\mathrm{succ}(x))$, $(U_k(x))_{k\in K}$ and $(U_k(\mathrm{succ}_i(x)))_{k\in K}$ as positive literals,
- $(\sharp,(s,K)) \in \left(\Sigma'^{\sharp}\right)^2$ such that m_i has a clause having exactly $Q_s(x)$ and $(U_k(x))_{k\in K}$ as positive literals,
- $((s,K),\sharp) \in \left(\Sigma'^{\sharp}\right)^2$ such that M_i has a clause having exactly $Q_s(x)$ and $(U_k(x))_{k\in K}$ as positive literals.

In other words, for each $i \in [d]$ we have

$$\psi_i = \bigvee_{((s,K),(s,K'))\in(\Sigma')^2} \begin{pmatrix} Q_s(x) \wedge Q_{s'}(\mathrm{succ}(x)) \wedge \\ \bigwedge\limits_{j\in K} U_j(x) \wedge \bigwedge\limits_{j\in K'} U_j(\mathrm{succ}_i(x)) \wedge \\ \bigwedge\limits_{j\in[k]\setminus K} U_j(x) \wedge \bigwedge\limits_{j\in[k]\setminus K'} U_j(\mathrm{succ}_i(x)) \end{pmatrix},$$

$$m_i = \bigvee_{(\sharp,(s,K))\in(\Sigma'^{\sharp})^2} \left(Q_s(x) \wedge \bigwedge_{j\in K} U_j(x) \wedge \bigwedge_{j\in[k]\setminus K} U_j(x) \right), \text{ and}$$

$$M_i = \bigvee_{((s,K),\sharp)\in(\Sigma'^{\sharp})^2} \left(Q_s(x) \wedge \bigwedge_{j\in K} U_j(x) \wedge \bigwedge_{j\in[k]\setminus K} U_j(x) \right).$$

Finally, define projection $\pi\colon \Sigma' \to \Sigma$ as $\pi((s,K)) = s$. The projection was defined by the formula and the tiling defined above expresses the same properties of the guessed predicates as does the formula, so we have

$$L = \pi(\mathcal{L}(\Delta_1,\ldots,\Delta_d))$$

and L is seen to be recognizable as a projection of a local language. □

We will collect the results of this section in the following corollary:

Corollary 3.3. *Given a d-language L, the following are equivalent:*

- $\mathrm{pixel}^d(L)$ *is definable in* EMSO
- $\mathrm{pixel}^d(L)$ *is definable in* $\mathrm{EMSO}(\forall^1)$
- $L \in \mathrm{REC}^d$.

4 Computational aspects of recognizable languages

Evaluating and otherwise examining word automata has many applications in computer science and real world programming because of the low complexity of such tasks—for pictures, it turns out the complexity is anything but usable. Below we will give examples of how the situation regarding computational aspects changes when moving from one dimensional words to pictures of even two dimensions.

4.1 Membership problem

Given a description of a word automaton and a word, it is trivial to check whether the automaton accepts the word or not. Given a description (in at least two-dimensions) of a tiling system and a picture, however, it is an NP-complete problem (see [Sch98]) to check whether the tiling system accepts the picture. In fact, there even exists a tiling system such that it is an NP-complete problem to ask whether a given picture is accepted by this fixed tiling system.

4.2 Emptiness problem

The emptiness problem for tiling systems is undecidable (see [GR92]). It is possible to encode a computation of a given Turing machine as a tiling system in two dimensions so that the input is placed on the first row and the further rows represent steps of the calculation. The language, then, includes pictures for which the computation halts before the last row is reached. However, the halting problem for Turing machines is known to be undecidable and it is now equal to asking whether the described picture language is empty.

4.3 Simulation of computation

In [Bor08] it is shown that the stopping calculations of of a non-deterministic Turing machine of complexity NPTIME(n^d) is recognizable language of $2d$-dimensional pictures. Another form of evidence of the computation complexity of tiling systems is also shown by the fact that a $(d+1)$-dimensional tiling system can simulate the calculation of a d-dimensional linear time non-deterministic cellular automaton, which again is a very powerful class of computations (see [GO16]).

4.4 Cellular automata

Those familiar with cellular automata will have noticed that the online tessellation automata introduced in section 2.4 is subfamily of cellular automata with the property that the scan is performed diagonally and a once determined (non-quiescent) value is never changed again, making the computation end after a linear number of steps. Another interesting fact related to recognizability is that the computation diagram of a d-dimensional cellular automaton is a recognizable $(d+1)$-dimensional picture, the required additional dimension being, of course, time.

The theory of picture languages has much more to do with cellular automata. Besides our logical characterization of REC studied here, in [GO16] the authors present another similar result which characterizes the linear running time of a d-dimensional non-deterministic cellular automata to a sublogic of existential second-order logic which allows $d+1$ universal quantifiers and predicate arity of $d+1$.

5 Discussion

While we have seen that the class of recognizable picture languages has many interesting characterizations that generalize those of word languages, it can also be seen that they are rather powerful ones when dimension is above one, and the computational complexity as examined in the previous section is much different than that of regular word languages. While these characterizations rose as a desire to find a characterization that unites logical, automata-based and tiling-based characterizations like in word languages, it may be that it is rather by the simplicity of one-dimensional case that so many classes of languages collapse into one and the same for word languages even though the computational model is simple.

Due to above, lately research has also shifted towards more limited versions of picture automata, tilings and logics in an attempt to find more computationally robust classes of picture languages that still would resemble regular word languages in some nice way. As far as the author knows, no comprehensive survey of these exist yet. One problem that seems common to these approaches is that the definitions tend to become much more difficult to understand: one attraction of regular word languages is that its characterizations based on regular expressions and automata so easy to state that undergraduate students can readily work with them.

As further work, creating a survey of the recent developments in recognizing pictures would surely be an interesting and fruitful journey. Also pictures, other concepts introduced here and even cellular automata generalize naturally for graphs. Indeed, before recognizable picture languages were defined, a corresponding tiling system for graphs was introduced by [Tho91], who also proved its equivalency with EMSO. Other closely related work also exists. For example [Kuu13] has studied messaging in distributed computing via message passing automata that is similar to cellular automata but operates on a directed graph and also introduces information hiding that limits the state information available to the neighbours or message receivers. Looking for more recent results among both generalized and specialized cases such as these to find transferable theorems not yet interpreted in one of them could also be a useful to further research.

References

[BH67] M. Blum and C. Hewitt. Automata on a 2-dimensional tape. In *8th Annual Symposium on Switching and Automata Theory (SWAT 1967)*, pages 155–160, Oct 1967.

[Bor08] Bernd Borchert. Formal language characterizations of P, NP, and PSPACE. *Journal of Automata, Languages and Combinatorics*, 13(3–4):161–183, 2008.

[Bü60] J. Richard Büchi. Weak second-order arithmetic and finite automata. *Mathematical Logic Quarterly*, 6(1-6):66–92, 1960.

[DG07] Arnaud Durand and Etienne Grandjean. First-order queries on structures of bounded degree are computable with constant delay. *ACM Trans. Comput. Logic*, 8(4), August 2007.

[Ebb95] Heinz-Dieter Ebbinghaus. *Finite model theory*. Perspectives in mathematical logic. Springer, Berlin, 1995.

[GO16] Etienne Grandjean and Frédéric Olive. A logical approach to locality in pictures languages. *Journal of Computer and System Sciences*, 82(6):959–1006, 2016.

[GR92] Dora Giammarresi and Antonio Restivo. Recognizable picture languages. *International Journal of Pattern Recognition and Artificial Intelligence*, 06(02n03):241–256, 1992.

[GR96] D. Giammarresi and A. Restivo. Two-dimensional finite state recognizability. *Fundamenta Informaticae*, 25(3-4):399–422, 1996. cited By 31.

[GR97] Dora Giammarresi and Antonio Restivo. Two-dimensional languages. In Grzegorz Rozenberg and Arto Salomaa, editors, *Handbook of Formal Languages: Volume 3 Beyond Words*, pages 215–267. Springer Berlin Heidelberg, Berlin, Heidelberg, 1997.

[GRST96] Dora Giammarresi, Antonio Restivo, Sebastian Seibert, and Wolfgang Thomas. Monadic second-order logic over rectangular pictures and recognizability by tiling systems. *Information and Computation*, 125(1):32–45, 1996.

[IN77] Katsushi Inoue and Akira Nakamura. Some properties of two-dimensional on-line tessellation acceptors. *Information Sciences*, 13(2):95–121, 1977.

[IT92] K Inoue and I Takanami. A characterization of recognizable picture languages. *Lecture Notes in Computer Science*, 654:133–143, 1992.

[KS11] Jarkko Kari and Ville Salo. A survey on picture-walking automata. In Werner Kuich and George Rahonis, editors, *Algebraic Foundations in Computer Science: Essays Dedicated to Symeon Bozapalidis on the Occasion of His Retirement*, pages 183–213, Berlin, Heidelberg, 2011. Springer Berlin Heidelberg.

[Kuu13] Antti Kuusisto. Modal Logic and Distributed Message Passing Automata. In Simona Ronchi Della Rocca, editor, *Computer Science Logic 2013 (CSL 2013)*, volume 23 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 452–468, Dagstuhl, Germany, 2013. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik.

[LS97] Michel Latteux and David Simplot. Recognizable picture languages and domino tiling. *Theoretical Computer Science*, 178(1):275 – 283, 1997.

[Mat07] Oliver Matz. Recognizable vs. regular picture languages. In *Computing and Informatics / Computers and Artificial Intelligence - CAI*, volume 4728, pages 112–121, 01 2007.

[RS59] Michael Rabin and Dana Scott. Finite automata and their decision problems. *IBM Journal of Research and Development*, 3:114–125, 04 1959.

[Sch98] N. Schweikardt. The monadic quantifier alternation hierarchy over grids and pictures. *Computer Science Logic*, 1414:441–460, 1998.

[Tho91] Wolfgang Thomas. On logics, tilings, and automata. In Javier Leach Albert, Burkhard Monien, and Mario Rodríguez Artalejo, editors, *Automata, Languages and Programming*, pages 441–454, Berlin, Heidelberg, 1991. Springer Berlin Heidelberg.